UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/779,328	02/13/2004	Umesh Madan	MS1-1851US	5250

22801 7590 01/25/2007

LEE & HAYES PLLC
421 W RIVERSIDE AVENUE SUITE 500
SPOKANE, WA 99201

EXAMINER	
TSAI, SHENG JEN	
ART UNIT	PAPER NUMBER
2186	

SHORTENED STATUTORY PERIOD OF RESPONSE	NOTIFICATION DATE	DELIVERY MODE
3 MONTHS	01/25/2007	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Notice of this Office communication was sent electronically on the above-indicated "Notification Date" and has a shortened statutory period for reply of 3 MONTHS from 01/25/2007.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

lhptoms@leehayes.com

Office Action Summary	Application No.	Applicant(s)	
	10/779,328	MADAN ET AL.	
	Examiner	Art Unit	
	Sheng-Jen Tsai	2186	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *29 November 2006*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-38* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-38* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *13 February 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office Action is taken in response to Applicants' Amendments and Remarks filed on November 29, 2006 regarding application 10,779,328 filed on February 13, 2004.

2. Claims 1, 8 and 27 have been amended.

Claims 1-38 are pending for consideration.

3. ***Response to Remarks and Amendments***

Applicants' amendments and remarks have been fully and carefully considered with examiner's response set forth below.

Amendments on Claims 1, 8 and 27

Applicant amends independent claims 1, 8 and 27 with the additional limitations of "wherein the new filter comprises at least one of a condition field, a data field, an expiration time field, a filter weight field, or a permanent flag field," "wherein the inverse query engine cache comprises at least one of an add filter module, a remove filter module, a matcher, a maintainer, a filter table, a most recently used list, or an expiration list," "removing a filter based on an expiration time," and "trimming the filter table."

It is noted that these newly added limitations raise question under 35 U.S.C 112, second paragraph due to the following reasons:

First, the added limitation of "wherein the new filter comprises at least one of a condition field, a data field, an expiration time field, a filter weight field, or a permanent flag field" requires that the new filter to posses at least one of five recited fields. Thus,

the new filter may simply contain only a data field and still meet the requirement without having an expiration time field.

Second, another added limitation recites "removing a filter based on an expiration time." However, as explained above, the currently amended claim does not require a new filter to have an expiration time field because having only one of the cited fields other than the expiration time field would meet the requirement. Thus, the limitation of "removing a filter based on an expiration time" lacks the supporting basis when the new filter does not have an expiration time field.

Further, it is noted that the previous cited reference (Schneider, US 5,668,987) teaches the added limitation of "wherein the new filter comprises at least one of a condition field, a data field, an expiration time field, a filter weight field, or a permanent flag field," because each filter contained in the filter tables [Tables T1 and T2, figure 3A, 310 and 320] has a data field.

Still, it is noted that Schneider also teaches the added limitation of "wherein the inverse query engine cache comprises at least one of an add filter module, a remove filter module, a matcher, a maintainer, a filter table, a most recently used list, or an expiration list," because the query engine [figure 2, 260] contains filter tables [Tables, figure 2, 250; Tables T1 and T2, figure 3A, 310 and 320].

Moreover, it is noted that Schneider also teaches the added limitation of "trimming the filter table," because Schneider teaches maintaining the inverse query engine cache at or below a maximum cache size as new items are added to the filter table [While the number of items is less than the cache size (step 401), the method

continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)] by trimming old, least used items from the table [generally cache rows are filled on a least-recently used (LRU) basis (column 7, lines 26-28)].

Although Schneider does not explicitly teach the limitation of "removing a filter based on an expiration time," a newly identified reference (Klein et al., US 6,631,374) does explicitly teach this element.

A new ground of claim analysis based on the previous cited reference (Schneider, US 5,668,987) in combination with a newly identified reference (Klein et al., US 6,631,374) has been made. Refer to the corresponding sections of the following claim analysis for details.

Remarks on Claim 17

Applicants contend that Schneider does not teach the limitation of "deducting the query size of each query removed from the cache size" and "adding a new query size to the cache size, the new query size identifying a size of the new query added to the inverse query engine cache." The Examiner disagrees with this assessment due to the following reasons:

First, Schneider teaches maintaining the inverse query engine cache at or below a maximum cache size as new items are added to the filter table [While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)] by trimming old, least used items from the table [the

method continues to execute and maintain the cache at maximum size; generally cache rows are filled on a least-recently used (LRU) basis (column 7, lines 26-28)]. Thus maintaining the inverse query engine cache at or below a maximum cache size includes both adding new entries and removing old entries.

Second, since Schneider's method continues to execute and maintain the cache at maximum size [(column 8, lines 26-29)], it must continuously monitor and keep track on the total size of cache, which requires the monitoring how many entries have been added and how many entries have been removed in order to update the total size.

Therefore, the Examiner's position regarding the merits of patentability of claim 17, and those claims depending from it, remains the same as stated in the previous Office Action.

Claim Rejections - 35 USC § 112

4. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

5. Claims 1-16 and 27-31 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Each of the independent claims 1, 8 and 27 is currently amended with the additional limitations of "wherein the new filter comprises at least one of a condition field, a data field, an expiration time field, a filter weight field, or a permanent flag field," "wherein the inverse query engine cache comprises at least one of an add filter module, a remove filter module, a matcher, a maintainer, a filter table, a most recently used list,

or an expiration list," "removing a filter based on an expiration time," and "trimming the filter table."

It is noted that these newly added limitations raise question under 35 U.S.C 112, second paragraph due to the following reasons:

First, the added limitation of "wherein the new filter comprises at least one of a condition field, a data field, an expiration time field, a filter weight field, or a permanent flag field" requires that the new filter to posses at least one of five recited fields. Thus, the new filter may simply contain only a data field and still meet the requirement without having an expiration time field.

Second, another added limitation recites "removing a filter based on an expiration time." However, as explained above, the currently amended claim does not require a new filter to have an expiration time field because having only one of the cited fields other than the expiration time field would meet the requirement. Thus, the limitation of "removing a filter based on an expiration time" lacks the supporting basis when the new filter does not have an expiration time field.

Claims 2-7, 9-16 and 28-31 are rejected by virtue of their dependency from claims 1, 8 and 27, respectively.

6. ***Claim Rejections - 35 USC § 102***

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

7. Claims 17-20, 22-24, 26, 32-33 and 37 are rejected under 35 U.S.C. 102(b) as being anticipated by Schneider (US 5,668,987).

As to claim 17, Schneider discloses **one or more computer-readable media** [main memory, figure 1A, 102; mass storage, figure 1A, 107] **storing computer-executable instructions** [program code, columns 13~66] **that, when executed on a computer** [central processor, figure 1A, 101], **perform the following steps**:

receiving a request to add a new query to an inverse query engine cache that stores multiple queries, each query having a query size associated therewith [After the cache is scanned and a "miss" occurs (i.e., no match), the subquery is executed and its input value(s)/result value pair is added to the top (i.e., first row) of the subquery cache (column 7, lines 28-33); figure 4A, step 401, "while items < cache rows" determines if the size of the items to be added would exceed the size of a cache row];

deriving a cache size that is a sum of query sizes of the queries stored in the inverse query engine [In particular, a subquery cache is provided having a size which can be dynamically adjusted by the system during execution of the query, for achieving an optimal cache size (abstract)];

determining if the cache size is at greater than or equal to a maximum cache size [While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)];

removing one or more queries from the inverse query engine cache if the cache size is greater than or equal to the maximum cache size [For the latter case (i.e., a cache employing multiple rows or entries), generally cache rows are filled on a least-recently used (LRU) basis (column 7, lines 26-28)];

deducting the query size of each query removed from the cache size [since the size of the cache is the sum of the size of all queries in the tables, whenever a query is removed from the cache, the size of the cache decreases by the size of the query];

adding the new query to the inverse query engine cache [While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29); First, Schneider teaches maintaining the inverse query engine cache at or below a maximum cache size as new items are added to the filter table [While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)] by trimming old, least used items from the table [the method continues to execute and maintain the cache at maximum size; generally cache rows are filled on a least-recently used (LRU) basis (column 7, lines 26-28)]. Thus maintaining the inverse query engine cache at or below a maximum cache size includes both adding new entries and removing old entries.

Second, since Schneider's method continues to execute and maintain the cache at maximum size [(column 8, lines 26-29)], it must continuously monitor and keep track on

the total size of cache, which requires the monitoring how many entries have been added and how many entries have been removed in order to update the total size]; **and adding a new query size to the cache size, the new query size identifying a size of the new query added to the inverse query engine cache** [since the size of the cache is the sum of the size of all queries in the tables, whenever a query is added to the cache, the size of the cache increases by the size of the query].

As to claim 18, Schneider teaches that **the method as recited in claim 17, wherein the removing step further comprises removing a query from the inverse query engine that has been used less recently than other queries stored in the inverse query engine cache** [generally cache rows are filled on a least-recently used (LRU) basis (column 7, lines 26-28)].

As to claim 19, refer to "As to claim 17" presented earlier in this Office Action.

As to claim 20, Schneider teaches that **the one or more computer-readable media as recited in claim 17, wherein the step of adding the new query size to the cache size is performed before determining if the cache size is greater than or equal to the maximum cache size** [figure 4A, step 401, "while items < cache rows" determines if the size of the items to be added would exceed the size of a cache row; While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)].

As to claim 22, Schneider teaches that **the one or more computer-readable media as recited in claim 17, wherein the new query size is received with the new**

query [figure 4A, step 401, "while items < cache rows" determines if the size of the items to be added would exceed the size of a cache row; While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)].

As to claim 23, Schneider teaches that **the one or more computer-readable media as recited in claim 17, further comprising instructions to perform the additional step of determining the new query size** [figure 4A, step 401, "while items < cache rows" determines if the size of the items to be added would exceed the size of a cache row; While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)].

As to claim 24, Schneider teaches that **the one or more computer-readable media as recited in claim 23, wherein the determining the new query size further comprises estimating the new query size** [figure 4A, step 401, "while items < cache rows" determines if the size of the items to be added would exceed the size of a cache row; While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)].

As to claim 26, Schneider teaches that **the one or more computer-readable media as recited in claim 17, wherein a query size is represented as a weight value that denotes the relative size of the query associated therewith with regard**

to other queries stored in the inverse query engine cache [the corresponding cache weight is the sum of the size of all filter items currently stored in the cache; While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)].

As to claim 32, Schneider teaches that **an inverse query engine having an integrated cache** [figure 2, 260 shows the inverse query engine; figure 3B shows a subquery cache (340) which is part of the query engine].

As to claim 33, Schneider teaches that **the inverse query engine as recited in claim 32, wherein the cache is bound to a finite size** [While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)].

As to claim 37, refer to "As to claim 18" presented earlier in this Office Action.

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 1-12, 14-15, 21, 27-31 and 34-36 are rejected under 35 U.S.C. 103(a) as being unpatentable over Schneider (US 5,668,987), and in view of Klein et al. (US 6,631,374, hereafter referred to as Klein).

As to claim 1, Schneider discloses **a method** [Database system with Subquery Optimizer (title)], **comprising**:

receiving a request to add a new filter [After the cache is scanned and a "miss" occurs (i.e., no match), the subquery is executed and its input value(s)/result value pair is added to the top (i.e., first row) of the subquery cache (column 7, lines 28-33)] **to a filter table** [figure 3A shows examples of filter tables TABLE T1 (310) and TABLE T2 (320)] **stored in an inverse query engine cache** [figure 2, 260 shows the inverse query engine; figure 3B shows a subquery cache (340) which is part of the query engine];

adding the new filter to the filter table [After the cache is scanned and a "miss" occurs (i.e., no match), the subquery is executed and its input value(s)/result value pair is added to the top (i.e., first row) of the subquery cache (column 7, lines 28-33)],

wherein the new filter comprises at least one of a condition field, a data field, an expiration time field, a filter weight field, or a permanent flag field [because each filter contained in the filter tables (Tables T1 and T2, figure 3A, 310 and 320) has a data field];

maintaining the inverse query engine cache at or below a maximum cache size [While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e.,

values) are added to the cache (column 8, lines 26-29)], **wherein the inverse query engine cache comprises at least one of an add filter module, a remove filter module, a matcher, a maintainer, a filter table, a most recently used list, or an expiration list** [because the query engine (figure 2, 260) contains filter tables (Tables, figure 2, 250; Tables T1 and T2, figure 3A, 310 and 320)];

removing a filter based on an expiration time [taught by Klein, see below];

trimming the filter table [because Schneider teaches maintaining the inverse query engine cache at or below a maximum cache size as new items are added to the filter table (While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)) by trimming old, least used items from the table (generally cache rows are filled on a least-recently used (LRU) basis (column 7, lines 26-28)]; **and**

wherein the inverse query engine cache is used exclusively by an inverse query engine to store filters associated therewith [Database system and methods are described for improving execution speed of database queries (e.g., for decision support) by optimizing execution of nested queries or "subqueries," such as are commonly used in client/server database environments. In particular, a subquery cache is provided having a size which can be dynamically adjusted by the system during execution of the query, for achieving an optimal cache size (abstract)].

Regarding claim 1, Schneider does not explicitly mention **removing a filter based on an expiration item.**

However, Schneider does teach removing a filter based on when the filter was used last time [generally cache rows are filled on a least-recently used (LRU) basis (column 7, lines 26-28)]. This provides a motivation for Schneider's invention to incorporate other removing scheme also based on an time interval.

Further, Klein teaches in the invention "System and Method for Providing Fine-Grained Temporal Database Access" a database access method where an expiration time field is associated with each memory block and the removing of the contents in the memory block is based on the expiration time field [Each extent is a block of contiguous memory locations which have an associated expiration time. The expiration time is used by a database storage manager to indicate a time after which the extent can be overwritten with new data. Preferably, the retention time of each transaction table entry does not exceed the expiration time. Write operations take precedence and will overwrite transaction table entries if necessary after the expiration time has elapsed. Overwritten transaction table entries will cause temporal access requests to fail for unavailable data (column 7, lines 11-20)].

Therefore, it would have been obvious for one of ordinary skills in the art to recognize the need to remove the contents of a memory block when the size of the memory is limited, as demonstrated by both Schneider and Klein, and that removing the contents of a memory block based on an expiration time is a straightforward and efficient way of achieving this goal, as demonstrated by Klein, and to incorporate it into the existing system disclosed by Schneider, to further enhances the cache

replacement strategy beyond the commonly adopted "Least Recently Used" scheme, and makes better utilization of the precious resource of cache capacity.

As to claim 2, Schneider teaches that **the method as recited in claim 1, further comprising maintaining the size of the inverse query engine cache between an optimal cache size and the maximum cache size** [In particular, a subquery cache is provided having a size which can be dynamically adjusted by the system during execution of the query, for achieving an optimal cache size (abstract); While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)].

As to claim 3, Schneider teaches that **the method as recited in claim 1, wherein the maintaining further comprises**:

determining if the addition of the new filter to the new filter table increases the cache size above the maximum cache size [figure 4A, step 401, "while items < cache rows" determines if the size of the items to be added would exceed the size of a cache row]; **and**

removing one or more filters from the filter table if the addition of the new filter causes the cache size to exceed the maximum cache size [For the latter case (i.e., a cache employing multiple rows or entries), generally cache rows are filled on a least-recently used (LRU) basis (column 7, lines 26-28)].

As to claim 4, Schneider teaches that **the method as recited in claim 3, wherein the determining step further comprises**:

determining a relative size of the new filter [figure 4A, step 401, "while items < cache rows" determines if the size of the items to be added would exceed the size of a cache row];

assigning a filter weight to the new filter based on the relative filter size [the corresponding weight is the size of the filter item; figure 4A, step 401, "while items < cache rows" determines if the size of the items to be added would exceed the size of a cache row];

deriving a cache weight by summing filter weights of all filters in the filter table including the new filter [the corresponding cache weight is the sum of the size of all filter items currently stored in the cache; While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)]; **and**

comparing the cache weight to the maximum cache size [After the cache is scanned and a "miss" occurs (i.e., no match), the subquery is executed and its input value(s)/result value pair is added to the top (i.e., first row) of the subquery cache (column 7, lines 28-33); While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)].

As to claim 5, Schneider teaches that **the method as recited in claim 1, wherein the maintaining further comprises**:

identifying a weight associated with the new filter [the corresponding weight is the size of the filter item; figure 4A, step 401, "while items < cache rows" determines if the size of the items to be added would exceed the size of a cache row];

adding the weight associated with the new filter to a cache weight that is the sum of filter weights of filters in the filter table, each filter having a filter weight [the corresponding cache weight is the sum of the size of all filter items currently stored in the cache; After the cache is scanned and a "miss" occurs (i.e., no match), the subquery is executed and its input value(s)/result value pair is added to the top (i.e., first row) of the subquery cache (column 7, lines 28-33); While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)]; **and**

comparing the cache weight to the maximum cache size [While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)].

As to claim 6, Klein teaches that **the method as recited in claim 1, further comprising identifying one or more expired filters in the filter table; and wherein the maintaining the inverse query engine cache further comprises removing at least one of the identified expired filters** [Each extent is a block of contiguous memory locations which have an associated expiration time. The expiration time is used by a database storage manager to indicate a time after which the extent can be

overwritten with new data. Preferably, the retention time of each transaction table entry does not exceed the expiration time. Write operations take precedence and will overwrite transaction table entries if necessary after the expiration time has elapsed. Overwritten transaction table entries will cause temporal access requests to fail for unavailable data (column 7, lines 11-20)].

As to claim 7, Schneider teaches that **the method as recited in claim 1, further comprising a least recently used filter in the filter table** [For the latter case (i.e., a cache employing multiple rows or entries), generally cache rows are filled on a least-recently used (LRU) basis (column 7, lines 26-28)]; **and**

wherein the maintaining the inverse query engine cache further comprises removing the least recently used filter from the filter table when a size of the inverse query engine cache reaches the maximum cache size [Finally, an "LRU Depth" of the deepest hit in the cache is also tracked. Since the cache can adjust its size dynamically, it itself has a "depth"--that is, how many rows comprise the cache at a given instance. An "LRU Depth" variable is therefore employed to track how deep (i.e., the deepest row) into the cache there has been a "hit" (column 2, lines 20-40)].

As to claim 8, Schneider teaches that **a system** [Database system with Subquery Optimizer (title)], **comprising**:

an inverse query engine [figure 2, 260 shows the inverse query engine] **configured to test an input against a collection of filters** [figure 3A shows examples of filter tables TABLE T1 (310) and TABLE T2 (320)];

cache associated with the inverse query engine [figure 3B shows a subquery cache (340) which is part of the query engine], **wherein the inverse query engine cache comprises at least one of an add filter module, a remove filter module, a matcher, a maintainer, a filter table, a most recently used list, or an expiration list** [because the query engine (figure 2, 260) contains filter tables (Tables, figure 2, 250; Tables T1 and T2, figure 3A, 310 and 320)];

the filter table stored in the cache and containing multiple filters [figure 3A shows examples of filter tables TABLE T1 (310) and TABLE T2 (320)], **wherein the new filter comprises at least one of a condition field, a data field, an expiration time field, a filter weight field, or a permanent flag field** [because each filter contained in the filter tables (Tables T1 and T2, figure 3A, 310 and 320) has a data field]; **and**

a maintainer configured to maintain a size of the filter table within definite cache bounds [While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)], **wherein the maintainer removes a filter based on an expiration time from the cache** [taught by Klein, see below] **and trim the cache** [Schneider teaches maintaining the inverse query engine cache at or below a maximum cache size as new items are added to the filter table (While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)) by trimming old, least used

items from the table (generally cache rows are filled on a least-recently used (LRU) basis (column 7, lines 26-28)].

Regarding claim 8, Schneider does not explicitly mention **removing a filter based on an expiration item.**

However, Schneider does teach removing a filter based on when the filter was used last time [generally cache rows are filled on a least-recently used (LRU) basis (column 7, lines 26-28)]. This provides a motivation for Schneider's invention to incorporate other removing scheme also based on an time interval.

Further, Klein teaches in the invention "System and Method for Providing Fine-Grained Temporal Database Access" a database access method where an expiration time field is associated with each memory block and the removing of the contents in the memory block is based on the expiration time field [Each extent is a block of contiguous memory locations which have an associated expiration time. The expiration time is used by a database storage manager to indicate a time after which the extent can be overwritten with new data. Preferably, the retention time of each transaction table entry does not exceed the expiration time. Write operations take precedence and will overwrite transaction table entries if necessary after the expiration time has elapsed. Overwritten transaction table entries will cause temporal access requests to fail for unavailable data (column 7, lines 11-20)].

Therefore, it would have been obvious for one of ordinary skills in the art to recognize the need to remove the contents of a memory block when the size of the memory is limited, as demonstrated by both Schneider and Klein, and that removing

the contents of a memory block based on an expiration time is a straightforward and efficient way of achieving this goal, as demonstrated by Klein, and to incorporate it into the existing system disclosed by Schneider, to further enhances the cache replacement strategy beyond the commonly adopted "Least Recently Used" scheme, and makes better utilization of the precious resource of cache capacity.

As to claim 9, Klein teaches that **the system as recited in claim 8, further comprising an expiration module configured to remove expired filters from the filter table** [Each extent is a block of contiguous memory locations which have an associated expiration time. The expiration time is used by a database storage manager to indicate a time after which the extent can be overwritten with new data. Preferably, the retention time of each transaction table entry does not exceed the expiration time. Write operations take precedence and will overwrite transaction table entries if necessary after the expiration time has elapsed. Overwritten transaction table entries will cause temporal access requests to fail for unavailable data (column 7, lines 11-20)].

As to claim 10, Klein teaches that **the system as recited in claim 9, wherein the maintainer is further configured to invoke the expiration module when a new filter is added to the filter table** [Each extent is a block of contiguous memory locations which have an associated expiration time. The expiration time is used by a database storage manager to indicate a time after which the extent can be overwritten with new data. Preferably, the retention time of each transaction table entry does not exceed the expiration time. Write operations take precedence and will overwrite

transaction table entries if necessary after the expiration time has elapsed. Overwritten transaction table entries will cause temporal access requests to fail for unavailable data (column 7, lines 11-20)].

As to claim 11, Schneider teaches that **the system as recited in claim 8, further comprising a trim module configured to remove one or more filters from the filter table when the cache reaches a maximum cache size** [While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29); Finally, an "LRU Depth" of the deepest hit in the cache is also tracked. Since the cache can adjust its size dynamically, it itself has a "depth"--that is, how many rows comprise the cache at a given instance. An "LRU Depth" variable is therefore employed to track how deep (i.e., the deepest row) into the cache there has been a "hit" (column 2, lines 20-40)].

As to claim 12, Schneider teaches that **the system as recited in claim 11, wherein the trim module is further configured to remove the one or more filters from the filter table until the cache is reduced to an optimal cache size** [In particular, a subquery cache is provided having a size which can be dynamically adjusted by the system during execution of the query, for achieving an optimal cache size (abstract); For the latter case (i.e., a cache employing multiple rows or entries), generally cache rows are filled on a least-recently used (LRU) basis (column 7, lines 26-28)].

As to claim 14, Schneider teaches that **the system as recited in claim 11, wherein**:

a filter weight is associated with each filter in the filter table [the corresponding weight is the size of the filter item; figure 4A, step 401, "while items < cache rows" determines if the size of the items to be added would exceed the size of a cache row]; **the cache size further comprises a cache weight that is a sum of all filter weights in the filter table** [the corresponding cache weight is the sum of the size of all filter items currently stored in the cache; While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)]; **the maximum cache size further comprises a maximum cache weight** [the corresponding cache weight is the sum of the size of all filter items currently stored in the cache; While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)]; **and wherein the trim module is further configured to deduct a filter weight from the cache weight when a filter associated with the filter weight is removed from the filter table** [For the latter case (i.e., a cache employing multiple rows or entries), generally cache rows are filled on a least-recently used (LRU) basis (column 7, lines 26-28); since the weight is the size, whenever a filter is removed from the cache, the size of the cache decreases by the size of the filter, so as the weight].

As to claim 15, Schneider teaches that **the system as recited in claim 14, further comprising a cache weight module configured to assign a filter weight to each filter added to the filter table, each filter weight identifying a relative size of a filter with regard to other filters in the filter table** [since the weight is the size, whenever a filter is added to the cache, the size of the cache increases by the size of the filter, so as the weight].

As to claim 21, refer to "A to claim 6."

As to claim 27, Schneider discloses **a method** [Database system with Subquery Optimizer (title)] **for maintaining an inverse query engine cache** [figure 2, 260 shows the inverse query engine; figure 3B shows a subquery cache (340) which is part of the query engine], **comprising**:

determining when inverse query engine cache usage is approaching a cache usage capacity [figure 4A, step 401, "while items < cache rows" determines if the size of the items to be added would exceed the size of a cache row; While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)], **wherein the cache usage capability is determined by a size of a filter table comprises at least one of a condition field, a data field, an expiration time field, a filter weight field, or a permanent flag field** [because each filter contained in the filter tables (Tables T1 and T2, figure 3A, 310 and 320) has a data field];

removing one or more filters from the inverse query engine cache when the cache is approaching the cache capacity until the cache usage is reduced to an optimal cache usage [For the latter case (i.e., a cache employing multiple rows or entries), generally cache rows are filled on a least-recently used (LRU) basis (column 7, lines 26-28); In particular, a subquery cache is provided having a size which can be dynamically adjusted by the system during execution of the query, for achieving an optimal cache size (abstract)],

wherein removing one or more filters comprises at least one of expiring [taught by Klein, see "As to claim 1" and "As to claim 8"]**or trimming the cache** [Schneider teaches maintaining the inverse query engine cache at or below a maximum cache size as new items are added to the filter table (While the number of items is less than the cache size (step 401), the method continues to execute and maintain the cache at maximum size; any unique items (i.e., values) are added to the cache (column 8, lines 26-29)) by trimming old, least used items from the table (generally cache rows are filled on a least-recently used (LRU) basis (column 7, lines 26-28)]; **and**

wherein an inverse query engine cache comprises at least one of an add filter module, a remove filter module, a matcher, a maintainer, a filter table, a most recently used list, or an expiration list [because the query engine (figure 2, 260) contains filter tables (Tables, figure 2, 250; Tables T1 and T2, figure 3A, 310 and 320)].

As to claim 28, refer to "As to claim 6" presented earlier in this Office Action.

As to claim 29, Klein teaches that **the method as recited in claim 28, wherein an expired filter is a filter that has been stored in the inverse query engine cache for a predefined period of time** [Each extent is a block of contiguous memory locations which have an associated expiration time. The expiration time is used by a database storage manager to indicate a time after which the extent can be overwritten with new data. Preferably, the retention time of each transaction table entry does not exceed the expiration time. Write operations take precedence and will overwrite transaction table entries if necessary after the expiration time has elapsed. Overwritten transaction table entries will cause temporal access requests to fail for unavailable data (column 7, lines 11-20)].

As to claim 30, refer to "As to claim 29" presented earlier in this Office Action.

As to claim 31, refer to "As to claim 7" presented earlier in this Office Action.

As to claim 34, refer to "As to claim 6" presented earlier in this Office Action.

As to claim 35, refer to "As to claim 29" presented earlier in this Office Action.

As to claim 36, refer to "As to claim 29" presented earlier in this Office Action.

10. Claims 6, 9-10, 13, 16, 21, 25, 28-30 34-36 and 38 are rejected under 35 U.S.C. 103(a) as being unpatentable over Schneider (US 5,668,987), and in view of Minami et al. (US Patent Application Publication 2003/0165160).

As to claims 6 and 9-10, Schneider does not teach **identifying one or more expired filters in the filter table; and removing at least one of the identified expired filters**.

However, Minami et al. teach in their invention "Gigabit Ethernet Adapter" the use of an ARP (Address Resolution Protocol) cache module where entries of the cache may expire [A static ARP cache entry is created in the ARP cache when the internal processor requests the ARP cache module create an ARP cache entry. The internal processor may also create dynamic ARP cache entries. A dynamic ARP cache entry exists for time specified by the user before the ARP cache entry expires, and the ARP cache module removes the cache entry. Expiration time for a dynamic ARP cache entry is typically five to 15 minutes. A static ARP cache entry does not normally expire (paragraph 0256)].

Identifying and removing an expired entry that is no longer needed further enhances the cache replacement strategy beyond the commonly adopted "Least Recently Used" scheme, and makes better utilization of the precious resource of cache capacity.

Therefore, it would have been obvious for one of ordinary skills in the art to recognize the benefits of Identifying and removing an expired entry that is no longer needed, as demonstrated by Minami et al., and to incorporate it into the existing system disclosed by Schneider, to further enhances the cache replacement strategy beyond the commonly adopted "Least Recently Used" scheme, and makes better utilization of the precious resource of cache capacity.

As to claim 13, Minami et al. teach **that the system as recited in claim 11, wherein the trim module is further configured to determine if a permanent flag in a filter is set and, if the permanent flag is set, to leave the filter in the filter table**

[The user may create a static ARP cache entry. A static ARP cache entry is normally permanent and does not expire (paragraph 0265)].

As to claim 21, refer to "As to claims 6 and 9-10" presented earlier in this Office Action.

As to claim 25, refer to "As to claim 13" presented earlier in this Office Action.

As to claim 28, refer to "As to claims 6 and 9-10" presented earlier in this Office Action.

As to claim 29, refer to "As to claims 6 and 9-10" presented earlier in this Office Action.

As to claim 30, refer to "As to claims 6 and 9-10" presented earlier in this Office Action.

As to claim 34, refer to "As to claims 6 and 9-10" presented earlier in this Office Action.

As to claim 35, refer to "As to claims 6 and 9-10" presented earlier in this Office Action.

As to claim 36, refer to "As to claims 6 and 9-10" presented earlier in this Office Action.

As to claim 38, refer to "As to claim 13" presented earlier in this Office Action.

11. Claim 16 is rejected under 35 U.S.C. 103(a) as being unpatentable over Schneider (US 5,668,987), and in view of Baum et al. (US 4,928,239).

As to claim 16, Schneider teaches using a "Least Recently Used" algorithm for replacing cache entries [For the latter case (i.e., a cache employing multiple rows or

entries), generally cache rows are filled on a least-recently used (LRU) basis (column 7, lines 26-28)], does not teach **a most recently used list**.

However, "most recently used" is exactly the opposite of "least recently used," thus it is obvious to one of ordinary skills in the art that, regarding replacing a cache entry, a "least recently used" entry should be the first one to be replaced while a "most recently used" entry should be the last one to be replaced.

Further, Baum et al. disclose in their invention "cache memory with Variable Fetch and Replacement Schemes" monitoring the "most recently used" entries for the purpose of cache replacement [according to the LRU replacement scheme, each time one of the blocks in the cache is accessed, it is marked most recently used and the LRU status bits of the others are adjusted accordingly (column 5, lines 5-8)].

Therefore, it would have been obvious for one of ordinary skills in the art to recognize that the "most recently used" is implied by "least recently used" as far as cache entry replacement is concerned, as demonstrated by Minami et al., hence lacking patentable significance.

12. ***Related Prior Art***

The following list of prior art is considered to be pertinent to applicant's invention, but not relied upon for claim analysis conducted above.

- DeMarcken et al., (US Patent Application Publication 2004/0249682), "Filling a Query for Travel Planning."
- Haas et al., (US 6,934,699), "System and Method for Loading a Cache with Query Results."

- Gupta et al., (US 7,035,846), "Method, Computer Programs and Apparatus for caching Directory Queries."
- Jackson, (US Patent Application Publication 2003/0123387), "Device and Method for Filtering Network Traffic."
- Bennett, (US Patent Application Publication 2003/0204664), "Cache with Multiway Steering and Modified Cycle Reuse."
- Fu et al., (US Patent Application Publication 2004/0111519), "Access Network Dynamic Firewall."

Conclusion

13. Claims 1-38 are rejected as explained above.

14. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

15. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Sheng-Jen Tsai whose telephone number is 571-272-4244. The examiner can normally be reached on 8:30 - 5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Matthew Kim can be reached on 571-272-4182. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Sheng-Jen Tsai
Examiner
Art Unit 2186

January 17, 2007

PRIMARY EXAMINER
1/21/07